The following is a list of possible questions for our quiz on January 28th. Some of the questions will not be asked in the quiz. All the questions that will appear in the quiz will appear exactly as shown below (however, numeric parameters may be changed). The quiz is closed textbook, closed notes and closed neighbors. Note that the questions, which did not appear in this quiz, still may appear in the exams. You will find a solution for these questions during lectures.

• **It is suggested that you bring your calculator** (you can use your calculator during the quiz on January 28th).

**#1:** Complete the following figure by specifying number systems we discussed in the classroom.

```
Arithmetic systems used in computers

Integers

Fractions

#2: Show “78_{10}” using unsigned integer format. How many bits are needed?

#3: What are the two major weaknesses in “unsigned integer”?

#4: Show the binary bit pattern for “-5” in 4-bit sign-magnitude integer. Show all you work.

#5: What are the two known problems about the sign-magnitude integers?

#6: What does the following 16-bit two’s complement number mean in the decimal format? Show your calculation.
What is “Overflow Flag” in a processor (how is it used) for handling operations on two’s complement integers?

Transform the following decimal number to the two’s complement binary number (using the 16-bit format: your processor is a 16-bit architecture processor): -63_{(10)}. Show all your work.

Which numbering system solves the two problems in the “sign magnitude integers”?

How will “0_{(10)}” be represented using n-bits two’s complement number?

How will “-1_{(10)}” be represented using n-bits two’s complement number?

Show the binary bit pattern for “-5” in 4-bit two’s complement integer.

What is the problem in representing fractions using the “fixed-point” method?

What does “floating point” in “floating point fraction numbers” mean?

What are the three components in the floating-point number expressions?

Show (by an example) how a floating-point number expression can represent a huge number.

Show (by an example) how a floating-point number expression can represent a tiny number.

What is “0.01011_{(2)}” in the decimal format? Show your work.

What is the binary bit pattern of +12_{(10)} in the bias of 127? Assume that the bit pattern consists of 8 bits. Show your work.

In IEEE-754 floating-point number standard, how “0” is represented?

In IEEE-754 floating-point number standard, if all the “exponent bits” are “1” (i.e., eight “1”’s), what can it mean?

Translate +262.75_{(10)} in the normalized binary format. Show your work.

What does “normalization” in IEEE-754 standard mean?

What is the advantage of using “normalized” format in IEEE-754 standard?

Show the binary bit pattern for +262.75_{(10)} in IEEE-754 format. Show your work.
**#26:** Show the binary bit patterns for the tiniest positive number in IEEE-754 floating point numbers.

**#27:** Show the binary bit patterns for the tiniest negative number in IEEE-754 floating point numbers (for the definition of “the tiniest negative”, use the definition in #25).

**#28:** Show the binary bit patterns for the largest positive number in IEEE-754 floating point numbers (for the definition of “the tiniest positive”, use the definition in #25).

**#29:** Show the binary bit patterns for the smallest negative number in IEEE-754 floating point numbers (for the definition of “the smallest negative”, use the definition in #26).

**Note:** For questions #15 through #17 above, the following figure will be provided in your quiz question sheet:

---

**#30:** What are the five different types of the numbers IEEE-754 floating-point numbers that can not be accurately represented by a processor?

**Note:** This question does not ask descriptions of the five different types of the numbers. Instead, it asks the name of the five different types of the numbers.

****** Programming Project #1 ***************

**#31:** What are “registers” in processors?

**#32:** How are registers in processors used when a binary executable program is executed?

**#33:** What is “PC-SPIM”?

**#34:** Why do we need “jr $31” at the end of an assembly program?
#35: What are “system calls” in “MIPS Simulator”?

#36: “li $t0, (1024)” is an illegal instruction (if you try to assemble that instruction using PC-SPIM simulator, that instruction will cause a syntax error). What’s wrong?

#37: What is the difference between “li $a0, 1024” and “la $a0, 1024” instructions? Assume that this computer system is a 32-bit system (i.e., all the registers are 32-bit registers and its ALU can deal with up to 32-bit inputs and outputs).

#38: “li $t1, $t0” is an illegal instruction (if you try to assemble that instruction using PC-SPIM simulator, that instruction will cause a syntax error). What’s wrong?